Entwurf eines Hilfesystems für Petrinetzmodellierer

Knut Pitschke, Olaf Schröder, Claus Möbus

Zusammenfassung

Es wird die Konzeption eines Hilfesystems vorgestellt, das Personen bei der Modellierung mit Petrinetzen auf verschiedenen Entwurfsebenen mit wissensstandsbezogenen Hilfen unterstützen soll. Das Hilfesystem soll Benutzern mit unterschiedlichem Erfahrungshintergrund zu jedem Zeitpunkt optimale, dem aktuellen Wissensstand des jeweiligen Benutzers angepaßte Hilfen zur Verfügung stellen. Unerfahrenen Benutzern soll es möglich sein, sich durch die Interaktion mit dem System in die Modellierung mit Petrinetzen einzuarbeiten und das hierfür erforderliche Wissen schrittweise zu erwerben und zu optimieren.

Der Wissenserwerbsprozeß soll in *zwei Phasen* erfolgen. Zum einen soll es den Benutzern möglich sein, durch *"freies", ungeleitetes Problemlösen* eigenständige Lösungsentwürfe zu vorgegebenen Aufgaben zu entwickeln und sich dabei von dem System individualisierte Lösungshilfen geben zu lassen. Dabei werden die Lösungsentwürfe des Benutzers mit Transformationsregeln untersucht. Nicht erkennbare Entwürfe werden mit einer Verifikationskomponente überprüft, und es werden ggf. neue Transformationsregeln abgeleitet. Zum anderen soll die Möglichkeit zu *regelgeleitetem, durch adaptive Transformationshilfen unterstütztem Problemlösen* bestehen. Die Transformationshilfen sind visuelle Repräsentationen der Transformationsregeln.

Die Generierung wissensstandsbezogener Lösungs- und Transformationshilfen setzt wissensstandsbezogene Transformationsregeln und somit detaillierte Annahmen über das aktuelle Wissen des Benutzers *(Benutzermodell)* voraus. Grundlage für diese Annahmen ist die über den Benutzer rechnerseitig erfaßbare Information und ihre Interpretation mit Hilfe differenzierter Hypothesen zum Wissenserwerbsprozeß. Solche Hypothesen werden auf Basis der IDL-SDL-Theorie ("impasse-driven learning - success-driven learning") sowie empirischer Einzeluntersuchungen gewonnen, bzw. in Vorarbeiten gewonnene Hypothesen werden weiterentwickelt. Vorarbeiten zur Analyse von Lösungsentwürfen, zur Generierung von Hilfen und zur Modellierung hilfegeleiteter Wissenserwerbsprozesse wurden in unserem DFG-Projekt ABSYNT für die Domäne des funktionalen Programmierens durchgeführt.

Konzeption des Hilfesystems

Das geplante Hilfesystem soll für den Benutzer wissensstandsangepaßte Informationen als Hilfen bereitstellen, ohne daß der Benutzer in seiner Freiheit, explorativ vorzugehen oder originelle Lösungsideen zu verwirklichen, zu stark eingeschränkt wird. Die Entwicklung wissensstandsangepaßter Hilfen setzt detaillierte Hypothesen über den jeweils aktuellen Wissensstand des Benutzers voraus. Diese müssen aus

den Systemein-/ausgaben, also dem Mensch-Maschine-Dialog, abgeleitet werden. Theoretische Grundlage für die Entwicklung solcher Hypothesen ist die IDL-SDL-Theorie. Danach wird der Wissenserwerbsprozeß als Wechselspiel zweier Teilprozesse aufgefaßt: *impasse-driven learning (IDL)* [van Lehn, 1988; 1990] und *success-driven learning (SDL)* [Rosenbloom & Newell, 1987; Wolff, 1987]. Demnach ist der Lernende bevorzugt in Impasse- oder Stocksituationen zur Informationsaufnahme oder zu aktiver Informationssuche bereit. Als Ergebnis wird neues Wissen erworben und die Stocksituation überwunden (IDL). Wird dagegen bereits erworbenes Wissen erfolgreich genutzt, so wird es [im Sinne von Anderson, 1986; Lewis, 1987] optimiert (SDL). Mit Hilfe der IDL-SDL-Theorie können Informationen über die Tätigkeiten des Benutzers wie die Abfolge von Lösungsschritten oder das Aufsuchen von Hilfen als Grundlage für die Bildung von Hypothesen über erworbenes sowie optimiertes Wissen dienen.

Im Gegenstandsbereich "Modellierung mit Petrinetzen" besteht das vom Benutzer zu erwerbende *Planungswissen* aus *Transformationswissen* (Wissen zur Überführung einer deklarativen Aufgabenstellung in Konstrukte der Zielsprache, hier Netzteile) und *Kontrollwissen* (Heuristiken zur Auswahl bei verschiedenen Transformationsmöglichkeiten). Der hilfegeleitete Erwerb von Transformations- und Kontrollwissen soll gesteuert durch das Hilfesystem in zwei Phasen verlaufen. In der *ersten Phase* entwickelt der Benutzer ungeleitet "freie" Netzentwürfe zu vorgegebenen, einfachen Aufgaben. Der Benutzer kann die Entwürfe oder Ausschnitte davon vom System untersuchen lassen, indem er Prüfhypothesen über die Korrektheit dieser Ausschnitte formuliert. Der Benutzer erhält dann ggf. wissensstandsbezogene Lösungshilfen. In dieser Phase wird der Lernende mit den Netzkonstrukten vertraut gemacht und erwirbt implizit Transformationswissen. Das System entwickelt hierbei bereits Hypothesen über dieses vom Benutzer erworbene und verwendete Wissen. In der *zweiten Phase* erhält der Problemlöser Hilfen zur schrittweisen Transformation der gestellten Aufgabe in ein Netz. Durch dieses regelgeleitete Problemlösen können die Hypothesen über das Transformationswissen des Benutzers überprüft werden. Der Lernende erwirbt in dieser Phase explizit Transformationswissen und implizit Kontrollwissen.

Das Hilfesystem soll für die Unterstützung den beiden Problemlösephase aus folgenden Komponenten bestehen:

- einem *Expertenmodell*, das aus Transformationsregeln besteht und einen möglichst großen Lösungsraum aufspannt.
- einem *Benutzermodell* zur Repräsentation des jeweils aktuellen Wissensstands des Benutzers. Seine Bestandteile können sein: eine Teilmenge des Expertenmodells, fehlerhafte Transformationsregeln (Malrules), optimierte Regeln (Komposita) und Kontrollwissen. Das Benutzermodell wird anhand der Transformationsschritte des Benutzers kontinuierlich aktualisiert. Es dient der effizienten Diagnose von Lösungsentwürfen und der Generierung wissensstandsadäquater Hilfen.
- einer *Verifikationskomponente*, mit der Netzentwürfe im Hinblick auf die vorliegende Aufgabenstellung überprüft werden können.
- einem *Editor*, in dem Aufgaben präsentiert, Netze sowie Zwischenrepräsentationen konstruiert werden, und in dem Prüfhypothesen formuliert werden können.
- einer *Rückmeldungs- und Hilfekomponente* zur Präsentation der Systemantworten (wissensstandsangepaßte Lösungshilfen) nach Prüfhypothesen.
- einer Bibliothek wissensstandsangepaßter *Transformationshilfen.*

Diese Komponenten arbeiten wie folgt zusammen: In der ersten Phase entwickelt der Benutzer mit dem Netzeditor Lösungsentwürfe für die vorgelegten Aufgaben. Der Lernende kann die Entwürfe oder Teile davon vom System untersuchen lassen. Dazu übergibt er den gewünschten Entwurfsausschnitt dem System als Prüfhypothese. Dieser Entwurfsausschnitt wird dann mit den Transformationsregeln des Benutzermodells untersucht (vgl. [Möbus, Schröder, Thole, 1991] für die Domäne funktionalen Programmierens). Wenn nötig, werden zusätzlich Regeln des Expertenmodells zur Analyse herangezogen. Kann der Entwurfsausschnitt mit diesen Regeln erkannt werden, so gibt es zwei Möglichkeiten: Enthält die zum Parsen benutzte Regelmenge eine Malrule, so ist der Entwurf als inkorrekt erkannt. Anderenfalls ist er korrekt.

Nun kann der Benutzer Lösungshilfen in Form von Vervollständigungsvorschlägen und ggf. Korrekturvorschlägen von dem System anfordern. Diese Hilfen sind insoweit wissensstandsbezogen, als sie auf dem Benutzermodell beruhen, d.h. das Benutzermodell wählt aus verschiedenen Möglichkeiten der Vervollständigung eines Entwurfs eine geeignete aus. Außerdem wird das Benutzermodell aktualisiert: Die für die Analyse des Entwurfs benutzten Expertenmodell-Regeln werden in das Benutzermodell aufgenommen, und die benutzten Regeln des Benutzermodells werden durch Komposition optimiert. Diese Komposita befähigen das System einerseits, zukünftige, ähnliche Benutzerlösungen schneller zu erkennen, andererseits repräsentieren sie den Lernfortschritt des Benutzers.

Kann der Entwurfsausschnitt dagegen nicht erkannt werden, so kann er mit der Verifikationskomponente (z.B. dem Model checking Algorithmus, [Josko 1990; Damm, Döhmen, Gerstner, Josko, 1990]) weiter überprüft werden, wenn der Benutzer für seinen Entwurfsausschnitt ein Teilziel im Sinne der Aufgabenstellung angibt. Ausgangspunkte für die Verifikationskomponente ist eine geeignete, z.B. temporallogische Aufgabenspezifikation (s.u.).

Wird der Entwurfsausschnitt von der Verifikationskomponente als korrekt erkannt, so wird er als neues Kompositum dem Benutzermodell und dem Expertenmodell hinzugefügt. Im negativen Fall wird der Entwurfsausschnitt als neue Malrule [Sleeman, 1984] in das Benutzermodell aufgenommen. Die Malrules sollen neben der Unterstützung der Fehlererklärung das System befähigen, ähnliche Fehler künftig schneller zu erkennen.

In der zweiten Wissenserwerbsphase transformiert der Problemlöser unter Anleitung der Transformationshilfen die vom System geeignet präsentierten Aufgaben schrittweise in Netze. Die Transformationshilfen sind visuelle Repräsentationen der Transformationsregeln (inklusive Komposita) des Benutzermodells. Die konkreten, vom Schüler getroffenen Auswahlentscheidungen von Transformationshilfen gehen in das Benutzermodell ein. Die Transformationsentscheidungen geben Hinweise auf das Kontrollwissen. So lassen sich Präferenzen des Benutzers herausarbeiten (z.B. Präferenz der allgemeinsten, der speziellsten oder grundsätzlich der ersten angebotenen Regel).

Auf zwei Aspekte des beschriebenen Prozesses soll nun näher eingegangen werden:

- Die Entwicklung deklarativer Aufgabenbeschreibungen als Aufgabenstellungen für Benutzer sowie als Ausgangspunkt für die Ableitung von Lösungsentwürfen durch Transformationsregeln

- Die Entwicklung von Transformationsregeln (Expertenmodell) zur Diagnose von Lösungsentwürfen.

Entwicklung von Aufgabenbeschreibungen. Der erste Schritt in Richtung auf die Lösung eines beliebigen Problems besteht in seiner Formulierung. Wurde in ABSYNT die Aufgabenstellung noch in Form algebraischer Formeln präsentiert, so gestaltet sich die Formulierung der "typischen Petrinetzaufgabe" erheblich komplizierter. Dafür gibt es mehrere Gründe:

• In der bekannten Petrinetzliteratur [z.B. Baumgarten 1990; Peterson 1981; Reisig 1986] werden Netzbeispiele nicht als Lösungen vorgegebener Aufgaben präsentiert. Die Aufgabenstellung per se wird meist mit einem Satz wie z.B.: "Modellierung der Abläufe in einer typischen Bibliothek" beschrieben. Das Aussehen einer typischen Bibliothek bleibt in diesem Fall der Phantasie des Modellierers überlassen, während bei der praktischen Anwendung von Petrinetzen der Modellierer die internen Zusammenhänge durch Beobachtung und / oder Befragung am realen System erst erkennen muß.

• Für die verwendeten Beispielaufgaben muß das entsprechende reale System möglichst präzise, umfassend und detailliert, aber auch allgemeinverständlich beschrieben werden. Der Allgemeinverständlichkeit kann dabei nur durch eine umgangssprachliche Formulierung Rechnung getragen werden. Diese ist allerdings nicht präzise genug. Deswegen werden die zu modellierenden Abläufe und Zusammenhänge auch durch temporallogische Formeln beschrieben ([Kröger 1987] und [Josko 1990]). Diese Formeln werden von der Verifikationskomponente benötigt.

Folgendes Beispiel soll die verschiedenen Beschreibungsformalismen vorstellen: Es soll ein typischer Restaurantbetrieb modelliert werden.

"Normalerweise schläft der Wirt. Wird er, z.B. durch einen Gast, geweckt, ist er bereit, Aufträge entgegenzunehmen. Danach gibt er die Bestellung in die Küche und legt sich wieder aufs Ohr. In der Küche werden die Speisen zubereitet. Ist das Essen fertig, wird der Wirt geweckt. Er ist nun bereit, die Speisen zu servieren. Nach getaner Arbeit hält er sein wohlverdientes Nickerchen."

Es bedeuten:
Ws : Wirt schläft
WbA : Wirt ist bereit zur Auftragsannahme
WbS : Wirt ist bereit zum Servieren
K : Küche hat Auftrag erhalten
Z : Die Speisen werden zubereitet
E : Das Essen ist fertig

Die Aufgabenbeschreibung sieht dann folgendermaßen aus:

$\neg(Ws \wedge WbA)$
$\neg(Ws \wedge WbS)$
$\neg(WbA \wedge WbS)$
$Ws \vee WbA \vee WbS$

$WbA \rightarrow \Diamond (Ws \wedge K)$
$K \rightarrow \Diamond Z$
$Z \rightarrow \Diamond E$
$E \wedge Ws \rightarrow \Diamond WbS$
$WbS \rightarrow \Diamond Ws$
$Ws \rightarrow \Diamond WbA$

(Es bedeuten: $\neg$ Negation; $\wedge$ Konjunktion; v Disjunktion; $\rightarrow$ Implikation; $\leftrightarrow$ Äquivalenz;
$\Diamond$ X: "Es gibt einen zukünftigen Zeitpunkt, zu dem X gilt")

Entwicklung von Transformationsregeln. Ausgangspunkt hierzu ist die in unserem Projekt ABSYNT entwickelte Diagnosekomponente, die für vorgegebene Aufgabenstellungen funktionale Programmentwürfe erkennt und generiert [Möbus, Thole, 1990; Möbus, 1991]. Sie basiert auf einer Ziel-Mittel-Relation (ZMR), die einen UND-ODER-Baum definiert. Sie läßt sich, wie [Hölldobler, Schneeberger, 1990] gezeigt haben, auch zur Lösung allgemeiner Planungsprobleme verwenden. Durch die ZMR wird das Aufgabenziel in Subziele (UND-Knoten) zerlegt, die Subziele werden weiter ausdifferenziert bis auf die Ebene der funktionalen Sprachkonstrukte. Zu jeder Zieldifferenzierung bzw. -realisierung gibt es viele verschiedene Alternativen (ODER-Knoten). Für 22 ABSYNT-Programmieraufgaben können selbst bei einer auf sechs beschränkten Programmbaumhöhe mehrere Millionen Lösungen erkannt oder generiert werden. Mit Hilfe der ZMR können Lösungen generiert *(Synthese)*, Entwürfe erkannt *(Analyse)* und Teilentwürfe vervollständigt *(Hypothesentest)* werden. Diese drei Leistungen sollen auch im Hilfesystem für Petrinetzmodellierer realisiert werden. Wir haben dazu alternativ zwei Möglichkeiten untersucht:

(a) Die Entwicklung einer ZMR für verschiedene Petrinetz-Beispiele, z.B. aus [Reisig, 1986; Baumgarten, 1990].

(b) Die Prüfung der Eignung logischer Grammatiken und hier speziell der Definiten Klauselgrammatik (DCG) von [Pereira, Warren, 1980] zur Analyse von Petrinetzen. Es konnte gezeigt werden, daß das Hypothesenprüfen als Constraint Satisfaction Problem (CSP) unter Verwendung des DCG-Ansatzes formulierbar ist. Damit können mit der DCG Netze erkannt, generiert und Teilnetze vervollständigt werden. Ein ähnlicher Ansatz wurde unabhängig davon jüngst in [Tanaka, 1991] publiziert.

Wie sich ferner zeigte, lassen sich beide Ansätze (a) und (b) ineinander überführen. Für das geplante Hilfesystem wollen wir diese Ansätze jedoch verallgemeinern und die bisherigen Ziele der ZMR bzw. die Nichtterminale der DCG durch temporallogische Beschreibungen im oben dargestellten Sinne ersetzen. Damit können aufgabenübergreifende Transformationsregeln entwickelt werden, mit denen die als temporallogische Formeln präsentierten Aufgaben - ggf. über Zwischenrepräsentationen - in Lösungsentwürfe transformiert werden können [Olderog, 1989; Partsch, 1990].

Ausblick

Es sollen vier weitere Aspekte des geplanten Hilfesystems angesprochen werden:

- Die Möglichkeit zur Bearbeitung "offener" Planungsprobleme

- Die Modellierung von Wissenserwerbsprozessen
- Evaluationsstudien
- Die Erweiterung um eine Optimierungskomponente

Offene Planungsprobleme. Neben "freiem" und regelgeleitetem Problemlösen ist auch die Bearbeitung *"offener"*, selbst gestellter bzw. nicht oder nur unvollständig spezifizierter Planungsprobleme vorgesehen. Bei der Modellierung natürlicher Systeme (Organisationen, Produktionsabläufe, Bürokommunikation etc.) dürften fast immer *"offene" Planungsprobleme* vorliegen. Diese können z.B. in Form einer fiktiven oder empirischen Zeitreihe von Beobachtungsdaten, sowie ggf. Kriterien für das zu entwerfende Netz zur Reproduktion dieser Zeitreihe vorliegen. Von Seiten des Benutzers ist hier primär heuristisches Kontrollwissen zur Steuerung der Transformationsschritte gefordert.

Modellierung von Wissenserwerbsprozessen. Das Benutzermodell ist als integrierter Bestandteil des Hilfesystems konzipiert. Es dient der online-Diagnose von Wissenstrukturen und der Generierung von Hilfen, wobei Restriktionen wie Antwortzeiten des Systems beachtet werden müssen. Das Benutzermodell soll daher durch ein Wissenserwerbsmodell ergänzt werden, das nicht nur die Wissensstrukturen des Benutzers zu verschiedenen Zeitpunkten repräsentiert, sondern darüber hinaus die möglichen *Gründe* für die *Veränderung* dieser Wissensstrukturen. Das Wissenserwerbsmodell sollte eine Obermenge der Vorhersagen des Benutzermodells vorhersagen. Wird z.B. das Benutzermodell durch Expertenmodell-Regeln erweitert, so sollte das Wissenserwerbsmodell nicht nur dieselben Wissensstrukturen wie das Benutzermodell, sondern zusätzlich für diese Problemlöse-Episoden entsprechende Impasses im Sinne des IDL vorhersagen. Das Wissenserwerbsmodell soll u.a. vorhersagen, in welchen Problemlösesituationen der Lernende auf welche Hilfen zugreift. Im Zusammenhang mit IDL sind auch emotionale Komponenten zu berücksichtigen, wie die Entstehung von Ärger, Streß oder Befriedigung und ihre Rückwirkung auf den Planungsprozeß [z.B. Spies, Hesse, 1986]. Vorläufer solcher Wissenserwerbsmodelle wurden bereits im Projekt ABSYNT entwickelt [Schröder, Kohnert 1989/90].

Evaluation des Hilfesystems: Wir wollen das Hilfesystem unter zwei Gesichtspunkten empirisch evaluieren:
1. Sind *Petrinetze* geeignete Planungswerkzeuge für Modellierer offener Planungsprobleme? Wieweit werden sie von den Benutzern akzeptiert? Für welche Einsatzbereiche? Fällt es Novizen leicht, sich einzuarbeiten? Gibt es diesbezüglich Unterschiede beim freien vs. regelgeleiteten Problemlösen? Diese Fragen sollen entweder durch den Einsatz des Hilfesystems im Feld oder durch einen experimentellen Vergleich mit anderen Planungswerkzeugen (z.B. SADT, SIMSCRIPT/SIMFACTORY, ...) untersucht werden.

2. Welche Bedeutung haben *adaptive, wissensstandsbezogene* vs. *nichtadaptive* Hilfen in Bezug auf Lernfortschritt und Akzeptanz durch den Benutzer? Auch hier wollen wir detaillierte Vorhersagen machen: Werden z.B. die Hilfen, die mit dem aktuellen Wissensstand übereinstimmen, häufiger benutzt als andere? Werden die Hilfen benutzt, die mit der aktuellen Kontrollstrategie übereinstimmen? Bewirken nichtadaptive Hilfen einen Strategienwechsel, wie z.B. vermehrte Analogienutzung?

Optimierungskomponente: Bei der Optimierungskomponente wird daran gedacht, den Benutzer zu *zielgerichteterem, effizienterem Problemlösen* anzuleiten. Dabei wird nach einer korrekten, regelgeleitet

ausgeführten Problemlösung der Weg des Benutzers über die gewählten Zwischenzustände zum Ziel untersucht. Gibt es eine Regel, die mehrere Zwischenzustände überspringt, wird dem Schüler diese Regel zusammen mit der von ihm gewählten, bedeutungsgleichen Sequenz vorgestellt. Er kann dann entscheiden, ob diese "Abkürzung" für ihn erstrebenswerte Vorteile bringt und er sein Problemlöseverhalten in diesem Punkt in Zukunft verändern wird.

Literatur:

Anderson, J.R., Knowledge Compilation: The General Learning Mechanism. In: Michalski, R.S., Carbonell, J.G., Mitchell, T.M., Machine Learning, Vol. II, Los Altos: Morgan Kaufman, 1986, 289-310

Baumgarten, B., Petri-Netze - Grundlagen und Anwendungen. BI Wissenschaftsverlag, 1990

Damm, W., Döhmen, G., Gerstner, V., Josko, B., Modular Verification of Petri Nets. The Temporal Logic Approach. In de Bakker, J.W., de Roever, W.P., Rozenberg (eds), Proceedings REX-Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness. Springer: Lecture Notes in Computer Science 430, 1990

Hölldobler, S., Schneeberger, J., A New Deductive Approach to Planning. New Generation Computing, 8, 1990, 225-244

Josko, B., Verifying the Correctness of AADL Modules using Model Checking. In: de Bakker, J.W., de Roever, W.P., Rozenberg (eds), Proceedings REX-Workshop on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness. Springer: Lecture Notes in Computer Science 430, 1990, 387-400

Kröger, F., Temporal Logic of Programs, Berlin: Springer, 1987

Lewis, C., Composition of Productions. In: Klahr, D., Langley, P., Neches, R. (eds), Production System Models of Learning and Development. Cambridge: MIT Press, 1987, 329-358

Möbus. C., The Relevance of Computational Models of Knowledge Acquisition for the Design of Helps in the Problem Solving Monitor ABSYNT. In: Lewis, R, Otsuki, S. (eds), Advanced Research on Computers in Education. Elsevier North Holland, 1991, 137-144

Möbus, C., Schröder, O., Thole, H.-J., Runtime Modelling the Novice-Expert Shift in Programming Skills on a Rule-Schema-Case-Continuum, paper to be presented at the Workshop W.4 "Modelling for Intelligent Interaction", 12th International Joint Conference on Artificial Intelligence, IJCAI-91, Darling Harbour, Sydney, Australia, August 24-25, 1991

Möbus, C., Thole, H.-J., Interactive Support for Planning Visual Programs in the Problem Solving Monitor ABSYNT: Giving Feedback to User Hypotheses on the Language Level. In: Norrie, D.H., Six, H.W., Computer Assisted Learning. Proceedings of the 3rd Int. Conf. on Computer Assisted Learning ICCAL, 1990, 36-49

Olderog, E.-R., Nets, Terms, and Formulas: Three Views of Concurrent Processes and Their Relationship. Universität Oldenburg, FB Informatik, 1989

Partsch, H.A., Specification and Transformation of Programs: A Formal Approach to Software Development. Berlin, Springer 1990

Pereira, F.C.N., Warren, D.H.D., Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Transition Networks. Artificial Intelligence, 13, 1980, 231-278

Peterson, J.L., Petri Net Theory and the Modeling of Systems, Englewood Cliffs: Prentice Hall, 1981

Reisig, W., Petrinetze - eine Einführung. Berlin: Springer, 1986

Rosenbloom, P., Newell, A., Learning by Chunking. In: Klahr, D., Langley, P., Neches, R. (eds), Production System Models of Learning and Development. Cambridge: MIT Press, 1987, 221-286

Schröder, O., Kohnert, K., Toward a Model of Instruction-Based Knowledge Acquisition: The Operational Knowledge for A Functional, Visual Programming Language. Journal of Artificial Intelligence in Education, 1, 1989/90, 105-128

Sleeman, D., An Attempt to Understand Students´ Understanding of Basic Algebra. Cognitive Science, 8, 1984, 387-412

Spies, K., Hesse, F.W., Interaktion von Emotion und Kognition. Psychologische Rundschau, 37, 1986, 75-90

Tanaka, T., Definite-Clause Set Grammars: A Formalism for Problem Solving, Journal of Logic Programming, 1991,10,1-17

van Lehn, K., Toward a theory of Impasse-Driven Learning. In: Mandl, H., Lesgold, A. (eds), Learning Issues for Intelligent Tutoring Systems. New York, Springer, 1988, 19-41

van Lehn, K., Mind Bugs: The Origins of Procedural Misconceptions. MIT Press, 1990

Wolff, J.G., Cognitive Development as Optimization. In: Bole, L. (ed), Computational Models of Learning. Berlin, Springer, 1987, 161-205

Adresse der Autoren: Universität Oldenburg, Fachbereich Informatik, Abt. Lehr-Lernsysteme, Postfach 2503, D-2900 Oldenburg. E-mail: Claus.Moebus@arbi.informatik.uni-oldenburg.de

Peter Gorny (Hrsg.)

Informatik und Schule 1991

Informatik: Wege zur Vielfalt beim Lehren und Lernen

GI-Fachtagung
Oldenburg, 7.-9. Oktober 1991

Proceedings

Springer-Verlag
Berlin Heidelberg New York London Paris
Tokyo Hong Kong Barcelona Budapest

Herausgeber
Peter Gorny
Carl von Ossietzky-Universität Oldenburg
FB 10 – Informatik
Postfach 2503, W-2900 Oldenburg

4. Fachtagung „Informatik und Schule",
veranstaltet vom Fachbereich 7 „Ausbildung und Beruf" der GI
und der Carl von Ossietzky-Universität Oldenburg

CR Subject Classification (1991): K.3.1, K.3.2

ISBN 3-540-54619-7 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-54619-7 Springer-Verlag New York Berlin Heidelberg

Satz: Reproduktionsfertige Vorlage vom Autor
Druck- u. Bindearbeiten: Weihert-Druck GmbH, Darmstadt
33/3140-543210 – Gedruckt auf säurefreiem Papier